trinean

minimized samples, maximized throughput

DropSense96 System Integration Manual and API Reference Guide

trinean

Revision History

Revision Date:
April 2010

Revision April 2010.

This manual is published by TRINEAN NV/SA.

Questions or comments regarding the content of this manual can be

directed to the address below or to your TRINEAN representative.

TRINEAN NV/SA

Dulle Grietlaan 17/3

B-9050 Gentbrugge, Belgium

Tel. 0032 9 2727535

Fax. 0032 9 2727539

info@trinean.com

www.trinean.com

©DropSense 96 and ©DropPlate16/96 are trademarks of TRINEAN NV/SA.

This document is the copyright of TRINEAN and must

not be copied or reproduced in any form without prior written consent.

TRINEAN reserves the right to make technical improvements to

this equipment and documentation without prior notice as part of a

continuous program of product development.

This manual supersedes all previous editions.

Warranty and Special Provisions Limited Warranty

TRINEAN products are fully guaranteed for one year against defects in parts, materials, and workmanship. Defective parts and materials will be replaced or, at the discretion of TRINEAN, repaired at no charge for a period of one year and labor required for such replacement or repair will be provided at no charge for a period of one year, provided that the products are utilized and maintained in accordance with the instructions in the applicable operating and servicing manual, and provided further that the products have not, as determined solely by TRINEAN, been subject to misuse or abuse by the Customer or other parties unrelated to TRINEAN. TRINEAN makes no warranty, expressed or implied, as to the fitness of any product for any particular purposes other than those purposes described in the applicable operating and servicing manual, nor does TRINEAN make any other warranty, whether expressed or implied, including merchantability, other than those appearing on the face hereof. Where TRINEAN guarantees any product, whether under this Warranty or as a matter of law, and there is a breach of such guarantee, the Customer's only and exclusive remedy shall be the replacement or repair of defective parts and materials, as described above. This shall be the limit of TRINEANS' liability. Furthermore, TRINEAN shall not be liable for incidental or consequential damages. Failure of the Customer to notify TRINEAN of a claimed defect by registered mail within thirty days of the discovery thereof shall constitute a waiver of any claim for breach of warranty.

When a product is required by TRINEAN to be installed by a TRINEAN engineer or technician, the period of this Warranty shall begin on the date of such installation, provided however, that any use of the product prior to such installation shall, at sole election of TRINEAN, void this Warranty.

When installation by TRINEAN personnel is not required, the period of this Warranty shall begin on the date of shipment from TRINEAN. The period of this Warranty shall begin as described above whether or not the product has been installed pursuant to a purchase order, and any trial period shall be deducted from the Warranty period that would otherwise apply under a subsequent placed purchase order for that product. Limitation of Liability. Notwithstanding anything to the contrary contained herein, the Seller (whether by reason of breach of warranty, breach of contract, tort, or otherwise), including without limitation under any indemnification provision contained herein, shall be limited to replacement of goods returned to TRINEAN which are shown to TRINEANS' reasonable satisfaction to have been nonconforming or to refund the purchase price, or, if not paid, to a credit amount of the purchase price therefore.

THE FOREGOING WARRANTIES ARE EXCLUSIVE AND A RE GIVEN AND ACCEPTED IN LIEU OF ANY AND ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, THE IMPLIED WARRANTY OF MECHANTABILITY AND THE IMPLIED WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE. NEITHER PARTY SHALL BE LIABLE TO THE OTHER FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES

TABLE OF CONTENTS

1 Introduction 7
2 Physical integration : overview 8
2.1 Compatibility with liquid handling robots 8
2.2 Mechanical references on the instrument 10
3 Software integration: overview and concepts 12
4 Software integration: details 15
4.1 Installation and initialisation of the integration. 15
4.2 Definition of the interface files 16
4.3 Overview of the commands 19
4.3.1 Constructor 20
4.3.2 Request Access 20
4.3.3 Release Access 21
4.3.4 Open Tray 21
4.3.5 Close Tray 21
4.3.6 Get Status 22
4.3.7 Define Experiment 22
4.3.8 Start Measurement 22
4.3.9 Abort Measurement 23
4.3.10 Get Results 23
4.3.11 Exit 23
5 Appendix 1: examples 24
5.1 Example 1 : C# program using DLL 24
5.2 Example2: script using executables 26
6 Appendix 2 : list with status code 28

1 Introduction

The Trinean DropSense is a novel plate reader, designed to read the Trinean DropPlate. This DropPlate is a microfluidic disposable, compatible with SBS standard, allowing the measurement of the UV/vis absorption spectrum of small liquid samples (typically 1-3uL). Applications are concentration measurements of biomolecules such as DNA or proteins and read-out of colorimetric assays. Throughout this manual, the terminology “DropPlate” and “DropFrame” are used to describe the microfluidic chips.

The DropSense instrument is operated by the DropQuant software, running on a host PC. This host PC communicates with the instrument using a USB cable. The system can operate in stand-alone mode, or integrated in automated lab systems.

This manual describes the integration of the DropSense instrument (including the DropQuant software) in automated lab setups, for example integration with liquid handling robots. This includes:

- Physical integration: description of the mechanical references for integration the instrument on the work bench.
- Software integration: the application programming interface (API) for controlling the DropQuant software from another programming or scripting environment.

Note that the user aspects of the DropQuant software are described in detail in the user manual, this includes settings up experiments and retrieving the measurement results. This manual concentrates purely on the integration aspects.

2 Physical integration : overview

The Trinean DropSense instrument is designed for integration with liquid handling robots:

- The disposable is designed for automatic sample dispensing by liquid handling robots:
  - dimensions are compatible with SBS standard (input wells at 9mm pitch)
  - self-filling of the sample in the microfluidic meander reduces evaporation, allowing longer storage times of the filled DropPlate without influencing the concentration measurements.
- The mechanical design of the disposable and the in/out tray of the instrument are compatible with the grippers of liquid handling robots
- The instrument has mechanical features for indexing the position of the instrument on the work bench.

2.1 Compatibility with liquid handling robots

The handling of the DropPlate in a liquid handling robot, and the filling of the reservoirs by such robots, is shown in figures 1 and 2.

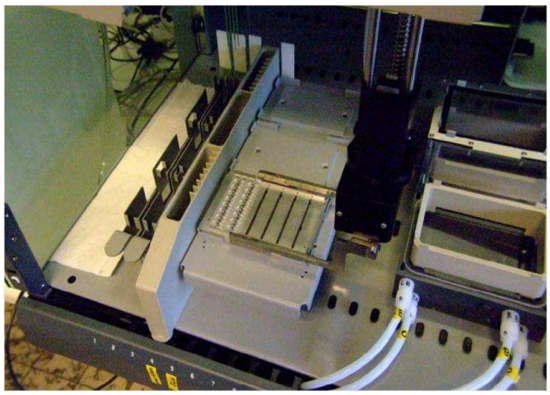

Figure 1 : liquid handling robots moving the DropFrame in the workbench.

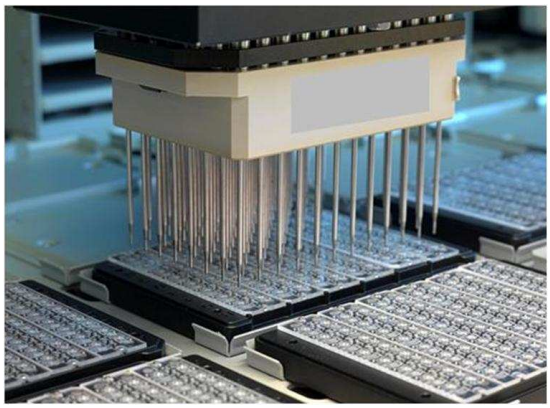

Figure 2 : A liquid handling robot filling the 96 input wells of the DropPlate.

The DropPlate is loaded into the instrument by the in/out tray. The in/out tray of the DropSense instrument allows pick up and put down by most common robotic arms. Note the alignment mark at the A1 corner of the frame: this defines the orientation of the DropPlate in the instrument.

The outer dimensions of the DropPlates are compatible with standard 96-well plates: width and length of the plate is 85.5x128mm. The height of the DropFrame is 10mm, and on top of the DropFrame the 1.4mm thick microfluidic disposable is mounted. Multiple variants of the DropPlate are available:

- An aluminum frame with six DP16 disposables mounted on the frame. The end-user has to mount the DP16 disposables on the aluminum frame (matching the hole-sleeve of the DP16 on the pins on the frame). In this case, the DropFrame includes 6 DropPlates.

- A moulded frame with the DP16 chips pre-assembled on the moulded frame (the end-user must not assemble the small DP16 disposable chips on the frame). In this situation, the DropPlate and the DropFrame refer to the same object.

All variants are compatible and interchangeable: outer dimensions and shape of the input reservoirs is the same.

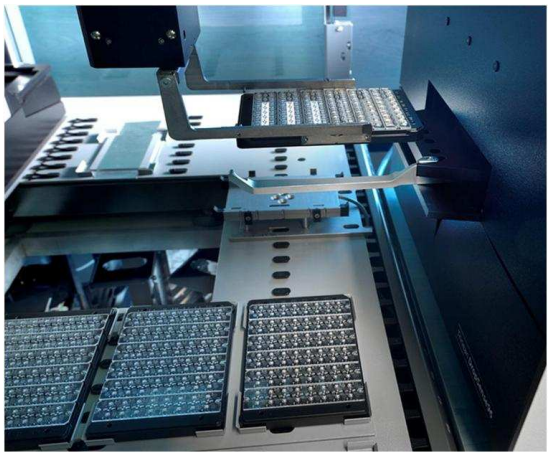

Figure 3 : robot arm placing the DropPlate on the in/out tray of the DropSense instrument.

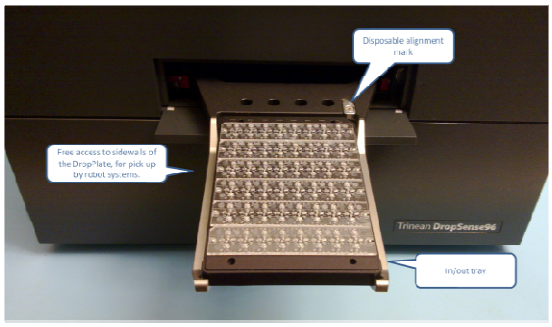
Disposable alignment
Free access to sidewalls of the DropPlate, for pick up by robot systems.
Trinean DropSense96
in/out tray

Figure 4 : DropPlate on the in/out tray of the DropSense instrument

The input wells are on same position as the input wells of a standard 96well plate (as defined in the SBS standard). The shape of the input well is shown in detail in figure 5.

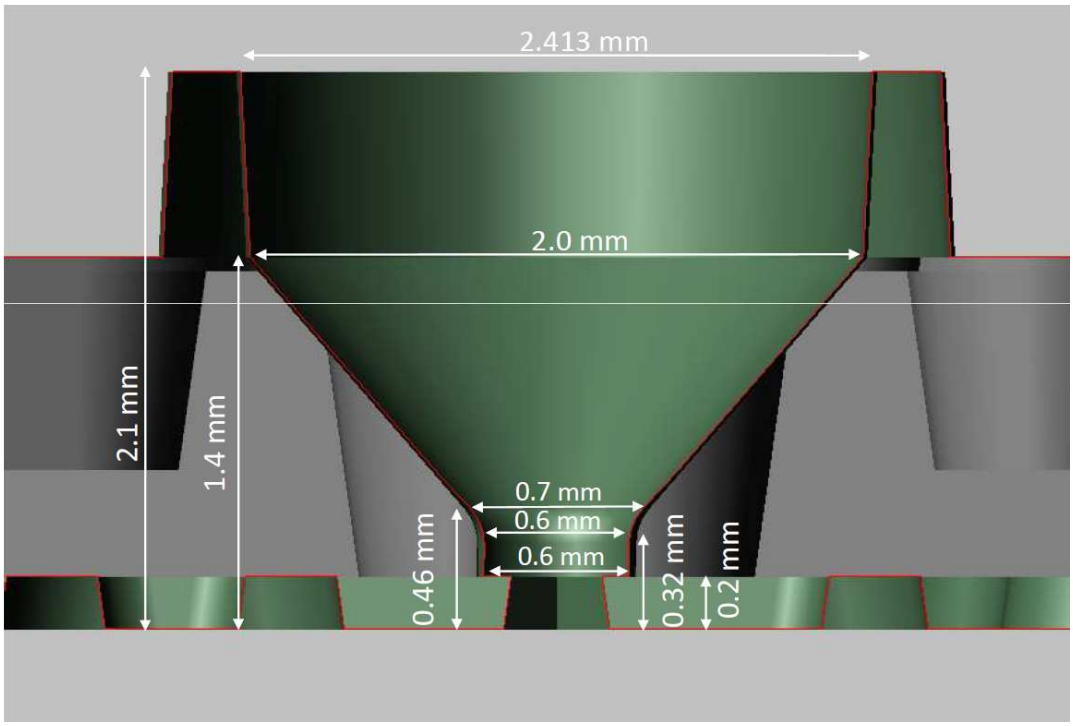

Figure 5 : dimensions of the input well.

Good dispensing of the sample in the input well requires:

- Avoid contact of the needle with the input well during dispensing.
- Bring the droplet in contact with the input well during dispensing.

With these guidelines, the 3uL sample will be transferred directly into the input meander. The height position of the input well relative to the ground plane is 11.4mm (10mm height of the DropFrame and 1.4mm thickness of the DP16).

The instrument includes an internal barcode scanner for reading the barcode. This requires that the barcode label is positioned at a specific location on the skirt of the DropFrame: the barcode is attached on the longest side of the plate, as shown in figure 6. The scanner can read CODE39 and CODE128 codes.

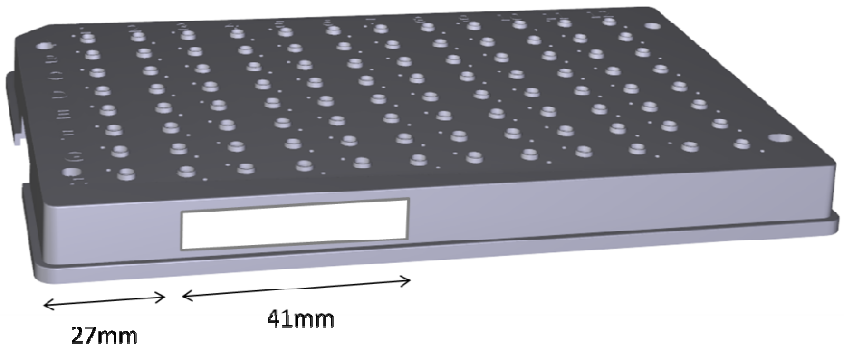

Figure 6 : definition of location for barcode that can be read by the internal barcode scanner in the instrument.

2.2 Mechanical references on the instrument

In most lab setups, the instrument will be mounted on the workbench of a liquid handling robot. When loading and unloading the DropFrame to and from the input tray, the robotic arm has to move the DropFrame to a specific (X, Y, Z) position relative to the instrument. The robot must know the exact position of the instrument, in order to put the DropPlate in the centre of the in/out tray.

To achieve this, there is a mechanical reference at the bottom side of the instrument, defining the exact position of the instrument on the workbench. This allows to define the (X, Y, Z) position of the DropFrame to put the DropFrame In the in/out tray.

The mechanical reference for positioning the DropSense instrument on the workbench consists of 2 10-mm holes, matching 10mm diameter pins. By placing the instrument on a plane with the 2 pins, the position of the instrument and the in/out tray is defined relative to these 2 pins.

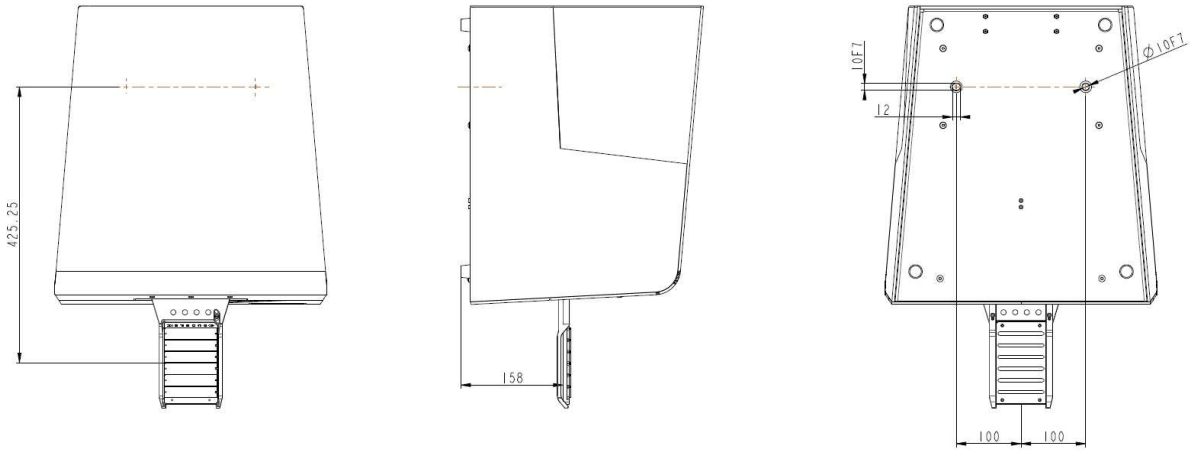

Figure 7 : features for defining the position of the in/out tray of the DropSense instrument : the indexing holes have 10mm diameter, the holes in the bottom plate are 12mm and act as a pre-alignment. The X, Y and Z coordinates of the DropPlate are defined on the figure.

If the tray is open, then the disposable must be positioned at next position relative to the two pins in the workbench: (see figure 7):

- Height: the ground plane of the 96-well disposable must be at 158mm relative to the reference plane.
- Front position: distance from the line between the two reference pins to the projection of the middle of the disposable on the reference plane is 425.25mm
- Lateral position: projection of the centre of the disposable on reference plane is in middle of the two reference pins.

3 Software integration: overview and concepts

The Trinean DropSense instrument is operated by the DropQuant software, running on a host PC. This host PC communicates with the instrument using a USB cable. The DropQuant software can be used in two modes:

- in local mode, the operator defines a measurement, starts and analyses the experiment with a few mouse clicks using the integrated graphical user interface. The use of the DropQuant software in local mode is explained in detail in the user manual.
- in remote mode (typically if the DropSense instrument is integrated with liquid handling robots in an automated lab environment), a script communicates with the DropQuant software and controls the measurement. Once the script is running, no further user interaction is required (for example: no need to click to “start button” to load a new disposable: instead the script sends commands to the DropQuant software to open and close the tray).

The user sets the operation mode of the DropQuant software, by choosing “remote” as login : if the software is in ‘remote mode’ the user interface is disabled and incoming messages will be processed. Trinean offers an “application programming interface” (API) for communication between a third party program or script with the DropQuant software. This manual describes the operation of the system in remote mode.

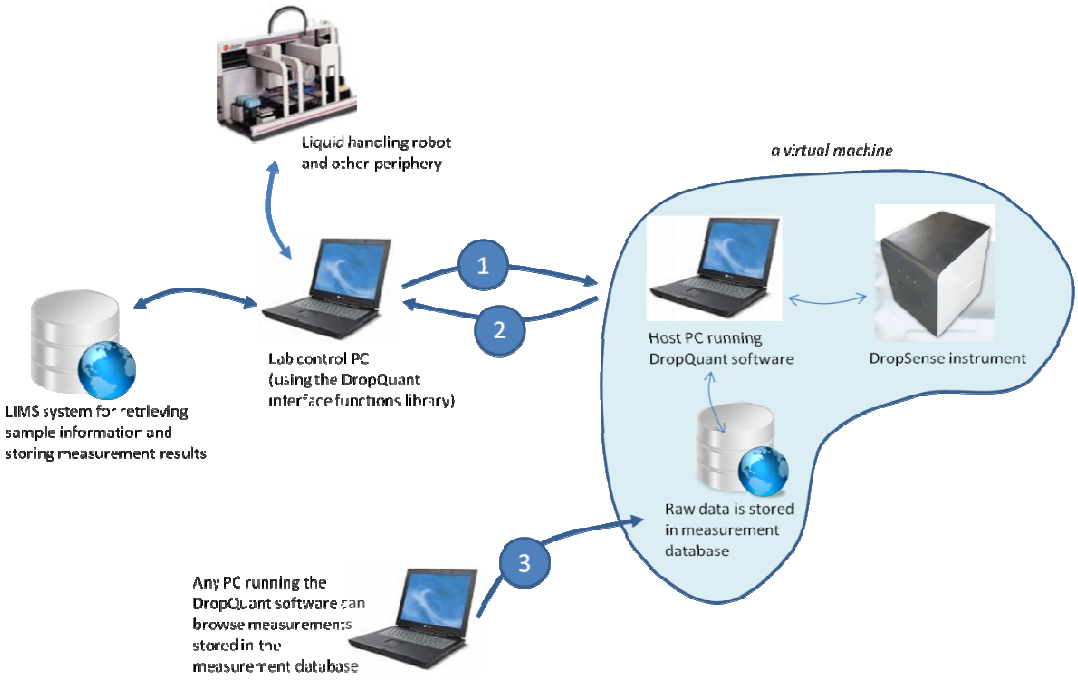
Liquid handling robot and other periphery
a virtual machine
1
2
3
Lab control PC (using the DropQuant interface functions library)
LIMS system for retrieving sample information and storing measurement results
Host PC running DropQuant software
DropSense instrument
Raw data is stored in measurement database
Any PC running the DropQuant software can browse measurements stored in the measurement database

Figure 8 : organisation of the measurement: (1) lab control PC asks the host PC to measure the plate, the host PC stores the raw data in the measurement database and (2) send the measurement results back to the lab control PC for further processing and/or storage of the results in customer LIMS system. (3) any PC with desired access rights can look up old measurements on the measurement database (for debugging, system maintenance, etc).

In remote mode, the DropQuant is controlled by a script running on the control PC (cfr figure 8). Note the script and the DropQuant software may run on the same PC, or on a separate PC connected by a network. The communication between script and controller program is organized over a TCP port. In case the script and DropQuant run on the same computer, this TCP port is bypassed.

The data is organized as follows:

- The user data is managed by the script on the control PC. This includes the name of the samples, source plates, concentrations, etc. The script generates the experiment settings, and processes the measurement results (for example, storing in his local LIMS system).
- The instrument data is stored on the host PC and not accessible by the script. This includes calibration data of the instrument, parameters of the disposables (such as path length of the reservoirs, pump profiles etc). There is no need to change this type of data, except if the system needs recalibration or if the information of a new type of disposables must be uploaded. Such interaction can be done in local mode only (requiring service interaction).
- The raw data of the measurement is stored in a user-defined path. The script can set the path for storing these raw data. This data includes all details of the measurement, including all spectra measured during pumping of the DNA in the measurement wells, etc.

The set of the DropSense instrument, DropQuant software and the location for storing the raw data can be seen as one “virtual instrument”.

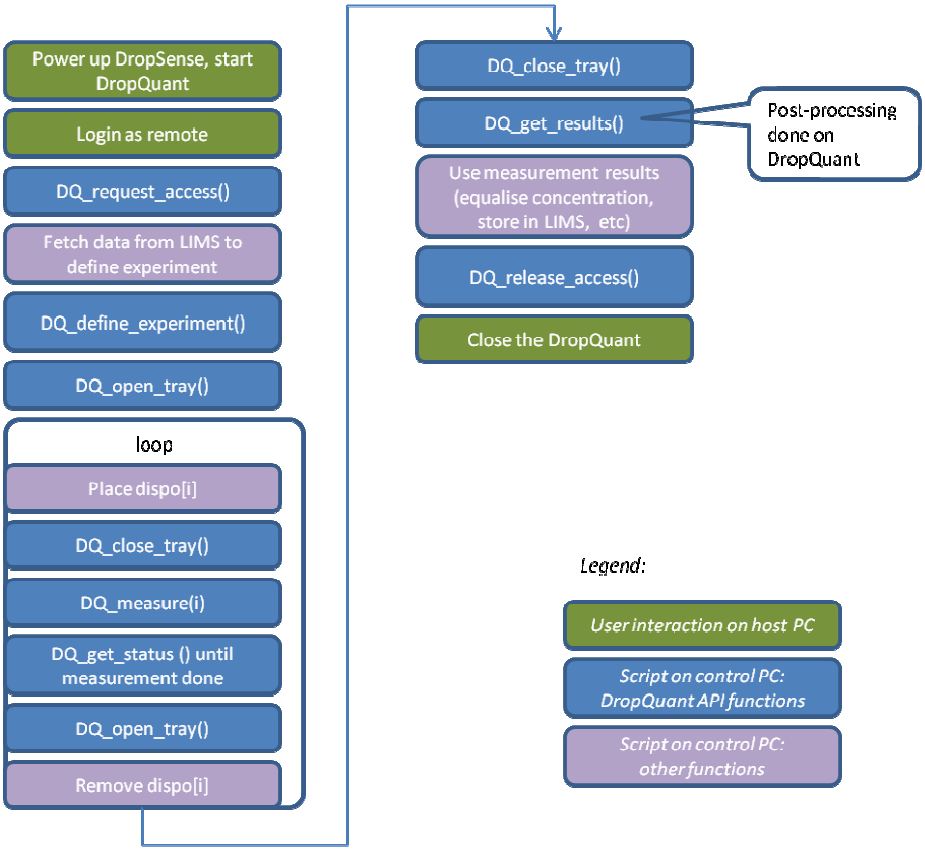

Figure 9 : example of a measurement algorithm

To perform a measurement, the control PC acts as a master and controls the host PC. The control PC asks the host PC to load/unload DropPlates, to read the plate in the DropSense instrument, and to send the measurement results back to the control PC. The control PC uses the results of the measurement (the concentration, OD, quality checks) for further processing (for example: communicate with the liquid handling robot for cherry-picking the good/bad samples from the plate, calculate the dilution factors and performing the dilution, etc).

We defined a complete set of functions allowing writing scripts and routines for controlling the instrument (initialization, loading and unloading a frame, starting a measurement, getting the status of the instrument). A typical algorithm is summarized in figure 7.

These functions are available through an API (application programming interface), and by executables. The executables have a limited functionality, and can be used in case the user script cannot access DLL functions. In case of executables, the communication (set parameters and returned re-sults) is organized via text files. The syntax of the executables is

```
DropQuant_Remote.exe –c command –a address:port –out output.txt
```

The functions from the DLL are executed in foreground: this implies the script is frozen during a call of a function, until the answer from the DropQuant is received indicating the command is executed successfully. Notable exception is the measure command: when starting a measurement the DropQuant controller sends an answer at the start of the measurement. During the measurement, the script may execute other tasks. The script can use the “get status” function to follow-up the status of the measurement, and take further actions as soon as the measurement of the plates is finished (for example deloading the old plate and loading a new plate). This is shown in figure 10.

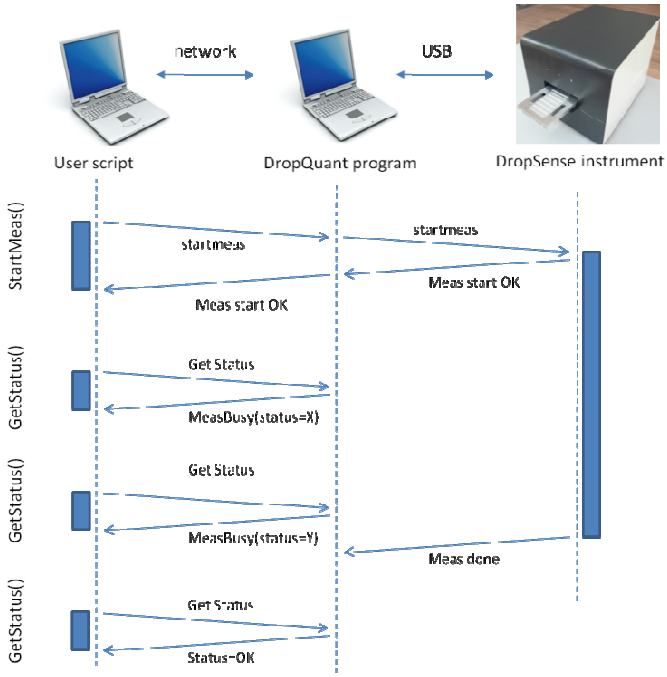

Figure 10 : communication during measurement

4 Software integration: details

4.1 Installation and initialisation of the integration.

The API is installed via a separate installer. After installation, next files are available on the system (default installation path is C:\Program Files\Trinean DropQuant\API):

- The DLL with the functions for communication with the DropQuant software.
- An executable `DropQuant_Remote.exe` : offer a simple interface in case the user script cannot access the DLL functions. Note this executable offers a reduced functionality compared to the DLL functions.
- Documentation and examples.

System requirements:

- The DropQuant program must be running on the host computer.
- There must be at least DropQuant user of type remote. When installing the DropQuant software, there is standard a login “remote” with empty password. The user with level “lab manager” can create more logins of type remote.
- A user of type “remote” must be created on the DropQuant software. This can be done by labmanager.
- The .NET environment must be installed on the control computer (version 2.0 or higher).
- A network connection between host PC and control PC (note host PC and controller PC can be same computer: then a TCP connection to ‘localhost’ is used.

Next start-up sequence must be followed to set up a remote connection:

1. Power-up the instrument and the controller PC
2. Connect the instrument to the controller PC (with USB cable)
3. Start up DropQuant program on the controller PC (=the communication over USB between DropSense and controller program is initialized).
4. Login as Remote user

After login as remote user, the screen of the DropQuant software looks like figure 10. At that moment, the DropQuant software is monitoring the TCP port for receiving commands.

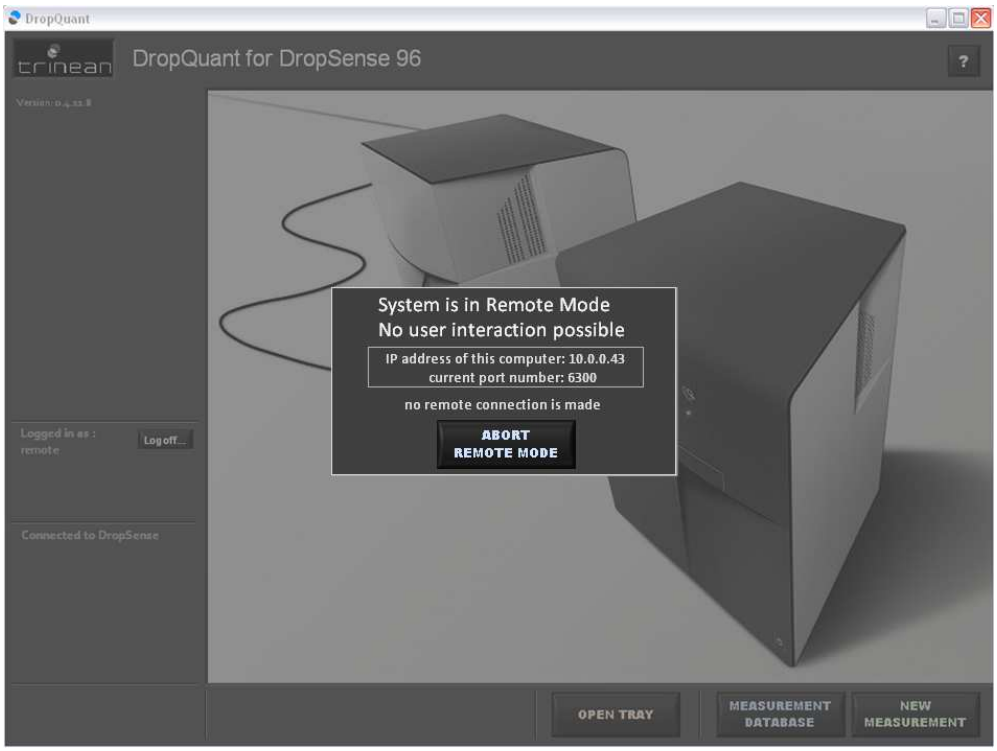

Figure 11: DropQuant screen when logged in as a remote user

The control PC making connection to the host PC needs the IP address of the host PC and the TCP port:

- The IP address can be found running the ipconfig command on the host PC. In case the script and the DropQuant program are running on the same PC, then the localhost keyword can be used.
- The standard value for the port is 6300[1].

Note bioth IP address and port number are visible on the user screen (see figure 11).

4.2 Definition of the interface files

To start a measurement, the host PC needs crucial information such as the number of samples, required results export format, etc. The control PC must send this information to the host PC. This is done via a number of text files or strings[2]. These are :

- Sample Definition String : text file or string describing the samples: position, name, reference to the dropframe, reference to the source plate, etc.
- Experiment Definition String : text file defining the settings of the experiment (such as name of the experiment, type of the disposable, etc)
- Results Definition String : text file defining the format of the results file.

The **Samples Definition String** is a text file with the definition of the samples. This file format is the same as used by the import feature of the DropQuant when operated in manual mode. It is a table with columns and rows. Each row represents one sample, in the columns the name, position,

[1] This can be changed in the DropQuant software from settings menu (requires service access)
[2] String in case of DLL functions, text file in case of exectuable

dropplate, type etc of the sample is defined. The columns may be random order: the Experiment Definition Settings file defines the meaning of the column (see next section).

An example of a sample definition string is shown in table 1:

```
DropFrame 1,A1,blank,DropFrame 1,A1
DropFrame 1,B1,sample1,DropFrame 1,A1
DropFrame 1,C1,sample2,DropFrame 1,A1
DropFrame 1,D1,sample3,DropFrame 1,A1
DropFrame 1,E1,sample4,DropFrame 1,A1
DropFrame 1,F1,sample5,DropFrame 1,A1
DropFrame 1,G1,sample6,DropFrame 1,A1
DropFrame 1,H1,sample7,DropFrame 1,A1
DropFrame 1,A2,sample8,DropFrame 1,A1
DropFrame 1,B2,sample9,DropFrame 1,A1
DropFrame 1,C2,sample10,DropFrame 1,A1
DropFrame 1,D2,sample11,DropFrame 1,A1
DropFrame 1,E2,sample12,DropFrame 1,A1
DropFrame 1,F2,ample13,DropFrame 1,A1
DropFrame 1,G2,sample14,DropFrame 1,A1
DropFrame 1,H2,sample15,DropFrame 1,A1
```

Table 1 : example Sample Definition String with16 samples on one plate, all samples using A1 as blank reference.

Note an experiment may contain multiple plates : all samples of the plates defined in the experiment are collected in one (large) sample definition string. After definition of the experiment, an array with the DropFrame_ID's defined in the sample definition string is returned. If the script tries to measure more plates than defined in the sample definition string, or if the script asks the DropQuant to send the results without having measured all plates defined in the Sample Definition File, then an error message is returned.

!	It is the responsibility of the script programmer to offer the exact number of plates to the DropSense instrument as defined in the sample definition string. After defining an experiment, the DropQuant software returns an array with names of the plates defined (cfr paragraph 4.3.7). This can be used in the script to make sure the plate inserted to the DropSense instrument are the ones that are expected by the DropQuant software.

Experiment Definition String contains 2 sections: [Experiment definition] and [Import samples]. The [Experiment definition]section describes the general settings of the experiment. The [Import samples] section describes the format of the sample definition file. Next fields must be in the [Experiment definition] section:

- Experiment name : user defined name[3]
- Experiment description : user defined text
- Sample type : "Nucleic Acids" or "Purified Proteins"
- Material : the material name. Note the name must be defined in the material database of the DropQuant program (installed at the host PC). Names that are defined standard are:
    - in case of Nucleic Acids : "dsDNA", "ssDNA", "RNA", "Oligo",

[3] If the string contains whitespaces, then add a " before and after the string.

  - in case of Purified Proteins : “General”, ”BSA”, “IgG” or “Multiple”
- Labeled = FALSE or TRUE
- (in case labeled=TRUE) label1 : name of the first label
- (in case labeled=TRUE) label2 : name of the second label (omit when only one label)
- Solvent = 0, 1 or 2 (0 = water based solvents; 1 = water + glycerol; 2 = water + DMSO)
- High viscosity = TRUE or FALSE
- Path length : one of “double pathlength” “single pathlength”
- Dropplate type : sets the type of disposable: one of “DP16-G” “DP16-U”.
- Use_barcode_check: use this option to active the internal barcode reader, this can be used as extra check to avoid error in order of inserting the DropPlates in the instrument.
- Measurement directory : path to (network) drive for storing the raw data of the measurement. Be sure to foresee sufficient disk space on this drive.

The [Import samples] section contains next fields:

- column_source_plate = column number containing the source plate ID (this is a reference to the plate in the biobank in which the extracted DNA is stored, typically this is a string with the plate name)
- column_source_position= column number containing the position (in A1 .. H12 notation)
- column_dropframe_ID= column number containing the ID of the DropFrame
- column_dropframe_position= column number containing the ..
- column_sample_name= column number containing the name
- column_blank_dropframe_ID= column number containing the ID of the frame with blank
- column_blank_dropframe_position= column number of the position of the blank.
- blanking_information : = 0=autoblank, 1=single blank (cfr 2 regeltjes hieronder), 2 = multiple blanks,zoals defined in de kolommen ?
- blank_dropframe_ID = definition of the frame ID of the blank in case of one blank for the complete experiment.
- blank_dropframe_position = definition of the blank position in case of one blank for the complete experiment. May be empty in case

An example is shown in table 1:

```
[Experiment definition]
experiment_name="Experiment by remote"
experiment_description="first DP16-U measurement"
sample_Type="Nucleic Acids"
material=dsDNA
labeled=TRUE
label1=”Cy3”
label2=”Cy5”
solvent=0
high_viscosity=FALSE
dropplate_type=DP16-U
pathlength="double pathlength"
measurement_directory = ”C:\Measurements”
use_barcode_check = TRUE

[Import samples]
column_source_plate=-1
column_source_position=-1
```

```
column_dropframe_ID=0
column_dropframe_position=1
column_sample_name=2
column_blank_dropframe_ID=3
column_blank_dropframe_position=4
blanking_information=2
blank_dropframe_ID=""
blank_dropframe_position=""
```

Table 2 : example of experiment definition string.

The **Results Definition String** is a text file defining the output format for the results. These are stored in a table-based text file.

This Results definition string contains a number of fields, followed by the column number. If the field should not be included in the results file, set the column number to -1.

- column_names : defines which columns need to be in the results (make a selection of the following columns: "Source Plate ID; Source Position; DropFrame ID; DropFrame Position; Sample name; Pump; Concentration (ng/µl); A260; A230; A280; A260/A230; A260/A280; Material; Date; Time; Conc.factor; Instrument ID; Pathlength mode; all OD values")
- separator : determines the character that will separate the different columns in the results (values that are allowed are ";" "," and "tab"

Note: if some fields are not defined in the sample definition file, then empty string will be in the results file.

```
[Export results]
column_names = "DropFrame ID;DropFrame Position;Sample
name;Pump;Concentration (ng/µl); A260;A280;A260/A280"
separator=;
```

Table 3 : example of results export format file.

4.3 Overview of the commands

Next is an overview of all functions defined in the API to interact with the DropQuant software.

All functions return a response to the script, with at least the status of the system after executing the requested command. Receiving the response from the system does not always mean that the system is ready for accepting new commands. For example, with DQ_Open_Tray(), the response is sent when the tray is open (after the action). After this, the system can execute new commands. However, with DQ_Start_Measurement(), the measurement is first initialized, and a response is sent if the measurement is successfully started or not. This way the script is not frozen during the measurement. The end of the measurement can be detected using the command DQ_Get_Status().

In case of using functions through the DLL, parameters are sent to the function using arguments. In case of the executables, the parameters are sent to the executable using text files, the path of the text files is part of the prompt. Thus the communication is through files.

To minimize the number of responses, return codes are used. Appendix 1 gives an overview of these return codes. If the command was successfully executed, 0 is returned. If not successfully executed, the reason of failure is found in the code, for example -1 if you don't have access because another remote user has access or the program is in 'local mode', -2 if there is no connection to the DropSense… If there is a need to give more details, but no error has occurred, positive number are used, for example if you want to determine the status of the ongoing measurement with DQ_GET_STATUS more specific information is required.

In case of executables, 3 arguments are always required:

`-c` command: there are a set of commands that can be used (f.e. –c opentray)
`-a` address and port: the address and port are given as address:port (f.e. –a localhost:6300)
`-out` the location of the file where the status code will be written to (f.e. –out "C:\DQ_result.txt")

All arguments may be in random order, but all are required before execution of the command.

4.3.1 Constructor

To create an object of the type DropQuant, include the parameters 'address' and 'port' to specify the communication parameters.

Syntax1:
```
DropQuant (String address, int port)
```

f.e. :
```
DropQuant DQ1 = new DropQuant(address, port);
```

Return code is a reference to the DropQuant (kind of handle).

Note there is no alternative function for the executable interface.

4.3.2 Request Access

Requests the access to the DropQuant program running on the host PC. If DropQuant is in Remote mode (logged in as a remote user) and no other computer has already access, then the access will be granted. (the Remote mode can be interrupted by a local user by clicking on "Abort Remote mode", no other actions can be taken by a local user)

Syntax1:
```
DQ1.DQ_Request_Access ()
```

Syntax2:
```
DQ.exe –c requestaccess
–a <address:port>
-out <output_path>
```

Return codes :

- status code 0 : successful
- Status code 1: access already active
- Status code -1 : no access (DropQuant is not in local mode, or DropQuant is connected to other computer)
- Status code -200 : could not connect over TCP

4.3.3 Release Access

Releases the access to the DropQuant program, so other remote users can request access. Note the DropQuant program remains in remote mode.

Syntax1: `DQ_Release_Access ()`

Syntax2:
```
–c releaseaccess
–a <address:port>
-out <output_path>
```

Return codes:

- status code 0 : success
- status code -1: no access
- status code -200 : could not connect over TCP

4.3.4 Open Tray

Opens the tray if the tray is not moving.

Syntax1: `DQ_Open_Tray ()`

Syntax2:
```
–c opentray
–a <address:port>
-out <output_path>
```

Return codes:

- status code 0 : success
- status code 3 : tray already open
- status code -1 : no access
- status code -2 : host computer not connected to DropSense
- status code -3 : could not open tray
- status code -52 : tray is moving
- status code -200 : could not connect over TCP

4.3.5 Close Tray

Closes the tray.

Syntax1: `DQ_Close_Tray ()`

Syntax2:
```
–c closetray
–a <address:port>
-out <output_path>
```

Return codes:

- status code 0 : success
- status code 4 : tray already closed
- status code -1 : no access
- status code -2 : host computer not connected to DropSense

- status code -52 : tray is moving
- status code -200 : could not connect over TCP

4.3.6 Get Status

This command can be send at all times and will return information about access, connection to DropSense, tray is moving/open/closed, measurement succeeded/failed/ongoing… If this command is executed during a measurement, extra information can be obtained.

Syntax1:
```
DQ_Get_Status(out String measurement_info)
```

Syntax2:
```
–c getstatus
–a <address:port>
-out <output_path>
```

Return: status code (see appendix 2)
+ measurement info

4.3.7 Define Experiment

Defines the experiment, based on two inputs: 'ExperimentDefinition' and 'SampleDefinition'. These two inputs have specific syntax and give information about experiment name, type of measurement, material, type of disposable, samples/blanks, sample names…

Syntax1:
```
DQ_Define_Experiment(String experimentdefinition,
String sampledefinition, out String [] dropframe_IDs)
```

Syntax2:
```
–c defineexperiment
–a <address:port>
-out <output_path>
-e <experiment_definition_path>
-s <sample_definition_path>
```

Return: status code (see appendix 2)
DropFrames : returns a string with the list of DropFrame ID's that are defined by the sample definition. This list can be used to start the measurement of the DropFrames.

Note: during measurement, the system assumes the plate are inserted in the instrument in the order defined in the dropframe_ID string.

4.3.8 Start Measurement

Starts the measurement, based on the DropFrame_ID. Every DropFrame should be started by a command. To know when the previous measurement is finished, use Get Status.

Syntax1:
```
DQ_Measure(String DropFrame_ID)
```

Syntax2:
```
–c measure
–a <address:port>
-out <output_path>
-d <dropframe_ID>
```

Return: status code (see appendix 2)

Note: if available in the system, the internal barcode scanner checks the DropFrame_ID.

4.3.9 Abort Measurement

Stops the measurement. The ongoing measurement will be interrupted by this command.

Syntax1: `DQ_Abort_Measurement ()`

Syntax2:
```
–c abort
–a <address:port>
-out <output_path>
```

Return: status code (see appendix 2)

4.3.10 Get Results

Returns the results of the measurement , ResultParameters determines what columns are in the report, what is used as separator and where the results can be stored (leave blank to choose the default path). If you enter a DropFrame_ID, only the results of that DropFrame will be given, if you leave it blank, all results will be given.

Syntax1:
```
DQ_Get_Results(String results_parameters, String dropFrame_ID,
out String results, out String experiment_path)
```

Syntax2:
```
–c getresults
–a <address:port>
-out <output_path>
-p <result_parameter_path>
-r <result_store_path>
[-d] <dropframe_ID>
```

`(-d is optional: if omitted, all results will be exported)`

4.3.11 Exit

Closes down the DropQuant program.

Syntax1: `DQ_Exit ()`

Syntax2:
```
–c exit
–a <address:port>
-out <output_path>
```

Return: status code (see appendix 2)

5 Appendix 1: examples

5.1 Example 1 : C# program using DLL

A simple example of the script is shown in next table (in C#)

```
using System;
using System.Collections.Generic;
using System.Text;
using DropQuant_Remote;

class Program
{
    static void Main(string[] args)
    {
        String address = "localhost";   //set the IP address
        int port = 6300;                // set the TCP port
        // add the experiment definition or read it from a file
        String exp_def =@"
[Experiment definition]
experiment_name=""Experiment by remote""
experiment_description=""example""
sample_Type=""Nucleic Acids""
material=dsDNA
labeled=FALSE
label1=""""
label2=""""
solvent=0
high_viscosity=FALSE
dropplate_type=DP16-U
pathlength=""double pathlength""
Measurement directory=""C:\DS_measurement""
[Import samples]
column_source_plate=-1
column_source_position=-1
column_dropframe_ID=0
column_dropframe_position=1
column_sample_name=2
column_blank_dropframe_ID=3
column_blank_dropframe_position=4
blanking_information=2
blank_dropframe_ID=""""
blank_dropframe_position="""""";
        // add the samples or read them from a file
        String samples = @"
DropFrame 1,A1,blank,DropFrame 1,A1
DropFrame 1,C1,sample1,DropFrame 1,A1
DropFrame 1,E1,sample2,DropFrame 1,A1
DropFrame 1,G1,sample3,DropFrame 1,A1
DropFrame 2,B1,sample4,DropFrame 1,A1
DropFrame 2,D1,sample5,DropFrame 1,A1
DropFrame 2,F1,sample6,DropFrame 1,A1
DropFrame 2,H1,sample7,DropFrame 1,A1";
        // add the result parameters or read them from a file
        String result_parameters = @"
[Export results]
```

```
column_names=""DropFrame ID;DropFrame Position;Sample
name;Pump;Concentration (ng/ul);A260;A280;A260/A280""
separator=;";
        String[] dropframes;
        int status;
        // create DropQuant object
        DropQuant DQ1 = new DropQuant(address, port);
        if (DQ1.DQ_Request_Access() < 0) { goto Exit; }
        if (DQ1.DQ_Define_Experiment(exp_def, samples, out dropframes) < 0)
            { goto End; }
        for (int i = 0; i < dropframes.Length; i++)
        {
            if (DQ1.DQ_Open_Tray() < 0) { goto End; }
            // load the DropFrame
            if (DQ1.DQ_Close_Tray() < 0) { goto End; }
            if (DQ1.DQ_Measure(dropframes[i]) < 0) { goto End; }
            do
            {
                status = DQ1.DQ_Get_Status();
                if (status != 32 & status != 25 & status != 31)
                    { goto End; }
            }
            while (status != 32 & status != 25);
        }
        String results, experiment_path;
        do
        {
            status = DQ1.DQ_Get_Results(result_parameters, "", out results,
                                                    out experiment_path);
            if (status == 0)
            {
                //write results to the location you want
                break;
            }
        } while (status == -104);
      // -104 is send when not all dropframes are processed yet
    End:
        status = DQ1.DQ_Release_Access();
    Exit:
    }
}
```

Next table is an example of the output file in text format:

```
DropFrame ID;DropFrame Position;Sample name;Pump;Concentration
(ng/µl);A260;A280;A260/A280
DropFrame 1;A1;blank;;0.00;0.00000;0.00000;-
DropFrame 1;C1;sample1;;277.59;5.55176;3.24380;1.71
DropFrame 1;E1;sample2;;281.40;5.62798;3.30268;1.70
DropFrame 1;G1;sample3;;277.46;5.54911;3.26779;1.70
DropFrame 2;B1;sample4;;277.40;5.54803;3.24143;1.71
DropFrame 2;D1;sample5;;282.18;5.64369;3.30929;1.71
DropFrame 2;F1;sample6;;282.49;5.64985;3.27086;1.73
DropFrame 2;H1;sample7;;279.00;5.58007;3.29418;1.69
```

5.2 Example 2 : script using executables

An example script is defined in next table:

```
DropQuant_Remote.exe -c requestaccess -a localhost:6300 -out c:\result.txt

@for /f %%i in (c:\result.txt) do @if %%i==0 ( @echo %%i ) else ( @echo %%i
& goto exit )

DropQuant_Remote.exe -c defineexperiment -a localhost:6300 -out
c:\result.txt -e c:\expdef.txt -s c:\samples.txt

@for /f %%i in (c:\result.txt) do @if %%i==0 ( @echo %%i ) else ( @echo %%i
& goto end )

DropQuant_Remote.exe -c opentray -a localhost:6300 -out c:\result.txt

@for /f %%i in (c:\result.txt) do @if %%i==0 ( @echo %%i ) else ( @echo %%i
& goto end )

DropQuant_Remote.exe -c closetray -a localhost:6300 -out c:\result.txt

@for /f %%i in (c:\result.txt) do @if %%i==0 ( @echo %%i ) else ( @echo %%i
& goto end )

DropQuant_Remote.exe -c measure -a localhost:6300 -out c:\result.txt -d
"DropFrame 1"

@for /f %%i in (c:\result.txt) do @if %%i==0 ( @echo %%i ) else ( @echo %%i
& goto end )

:statusloop

DropQuant_Remote.exe -c getstatus -a localhost:6300 -out c:\result.txt

@for /f %%i in (c:\result.txt) do @if %%i==31 ( @echo %%i & goto
statusloop) else ( @echo %%i )

DropQuant_Remote.exe -c opentray -a localhost:6300 -out c:\result.txt

@for /f %%i in (c:\result.txt) do @if %%i==0 ( @echo %%i ) else ( @echo %%i
& goto end )

DropQuant_Remote.exe -c closetray -a localhost:6300 -out c:\result.txt

@for /f %%i in (c:\result.txt) do @if %%i==0 ( @echo %%i ) else ( @echo %%i
& goto end )

DropQuant_Remote.exe -c measure -a localhost:6300 -out c:\result.txt -d
"DropFrame 2"

@for /f %%i in (c:\result.txt) do @if %%i==0 ( @echo %%i ) else ( @echo %%i
& goto end )

:statusloop2

DropQuant_Remote.exe -c getstatus -a localhost:6300 -out c:\result.txt
```

```
@for /f %%i in (c:\result.txt) do @if %%i==31 ( @echo %%i & goto
statusloop2) else ( @echo %%i )

:resultsloop

DropQuant_Remote.exe -c getresults -a localhost:6300 -out c:\result.txt -r
c:\results.csv -p c:\resultparameters.txt

@for /f %%i in (c:\result.txt) do @if %%i==-104 ( @echo %%i & goto
resultsloop ) else ( @echo %%i )

:end

DropQuant_Remote.exe -c releaseaccess -a localhost:6300 -out c:\result.txt

@for /f %%i in (c:\result.txt) do @echo %%i

:exit
```

6 Appendix 2 : list with status code

0	Successful
-1	No access
1	Access already achieved
-2	Not connected to DropSense
-3	Could not open tray
3	Tray was already open
-4	Could not close tray
4	Tray was already closed
-5	Tray not in right position
-9	Error reading experiment definition
-10	Error reading samples
-11	No pump profiles found
-12	No pump profiles found
-13	No samples defined
-14	Timeout for going to open tray position
20	Access free
21	Access and connection, no measurement
-22	Access and connection, measurement failed to initialize
23	Access and connection, measurement started
-24	Access and connection, measurement failed
25	Access and connection, measurement succesfull
30	Access and connection, measurement initializing
31	Access and connection, measurement busy
-31	Could not perform command, measurement busy
32	Access and connection, load next DropFrame
50	Access and connection, tray is open
51	Access and connection, tray is closed
-52	Access and connection, could not perform command, tray is moving
52	Access and connection, tray is moving
-99	Timeout: DropQuant not responding
-100	Access interrupted by local user interface
-102	Incorrect argument used, DropFrame ID not valid
-103	Incorrect argument used, DropFrame ID is already measured
-104	Results can only be exported if all DropFrames are measured
-200	Could not connect over TCP
999	DropQuant Exit accepted